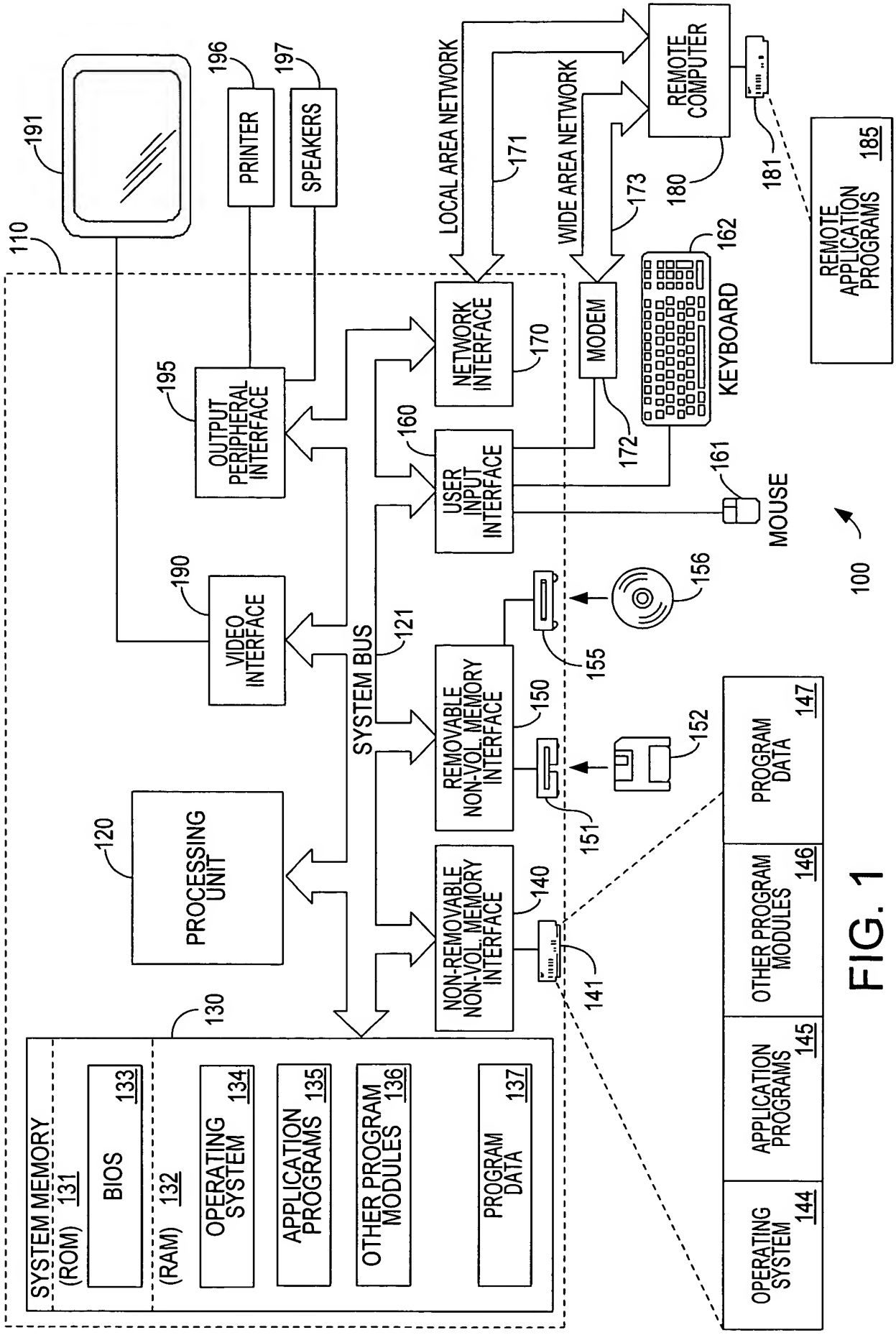

100
110
120
121
130
131
132
133
134
135
136
137
140
141
144
145
146
147
150
151
152
155
156
160
161
162
170
171
172
173
180
181
185
190
191
195
196
197
SYSTEM MEMORY
(ROM)
BIOS
(RAM)
OPERATING SYSTEM
APPLICATION PROGRAMS
OTHER PROGRAM MODULES
PROGRAM DATA
PROCESSING UNIT
SYSTEM BUS
VIDEO INTERFACE
OUTPUT PERIPHERAL INTERFACE
PRINTER
SPEAKERS
NETWORK INTERFACE
LOCAL AREA NETWORK
MODEM
WIDE AREA NETWORK
REMOTE COMPUTER
USER INPUT INTERFACE
KEYBOARD
MOUSE
REMOVABLE NON-VOL. MEMORY INTERFACE
NON-REMOVABLE NON-VOL. MEMORY INTERFACE
REMOTE APPLICATION PROGRAMS
OPERATING SYSTEM
APPLICATION PROGRAMS
OTHER PROGRAM MODULES
PROGRAM DATA

FIG. 1

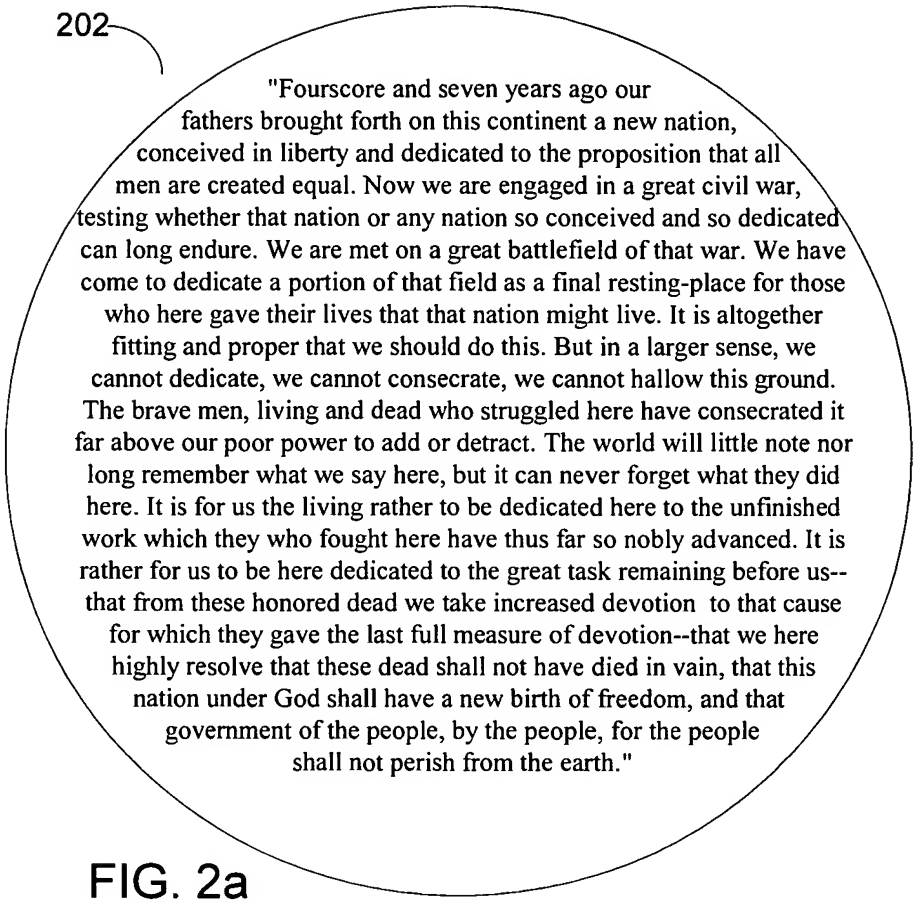


FIG. 2a

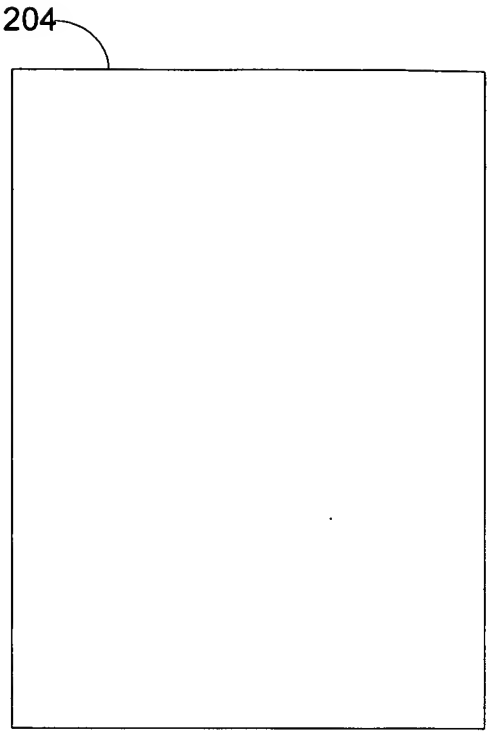
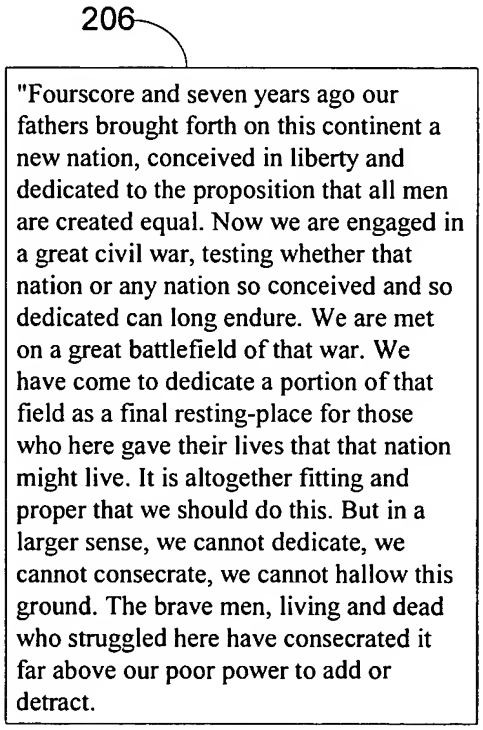


FIG. 2b



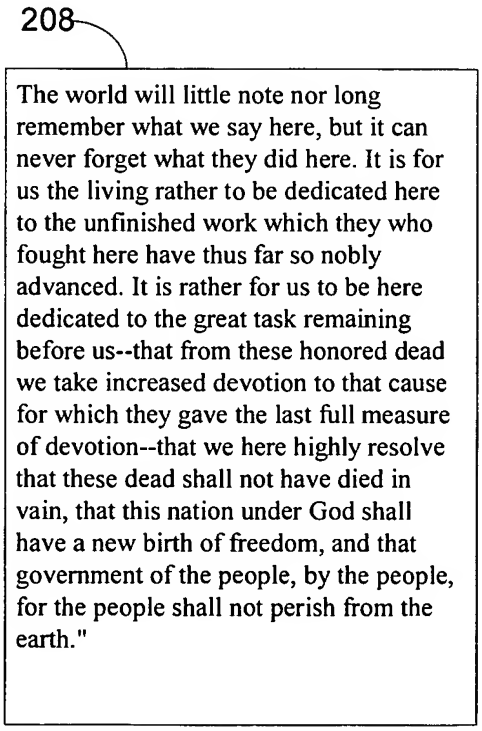


FIG. 2c

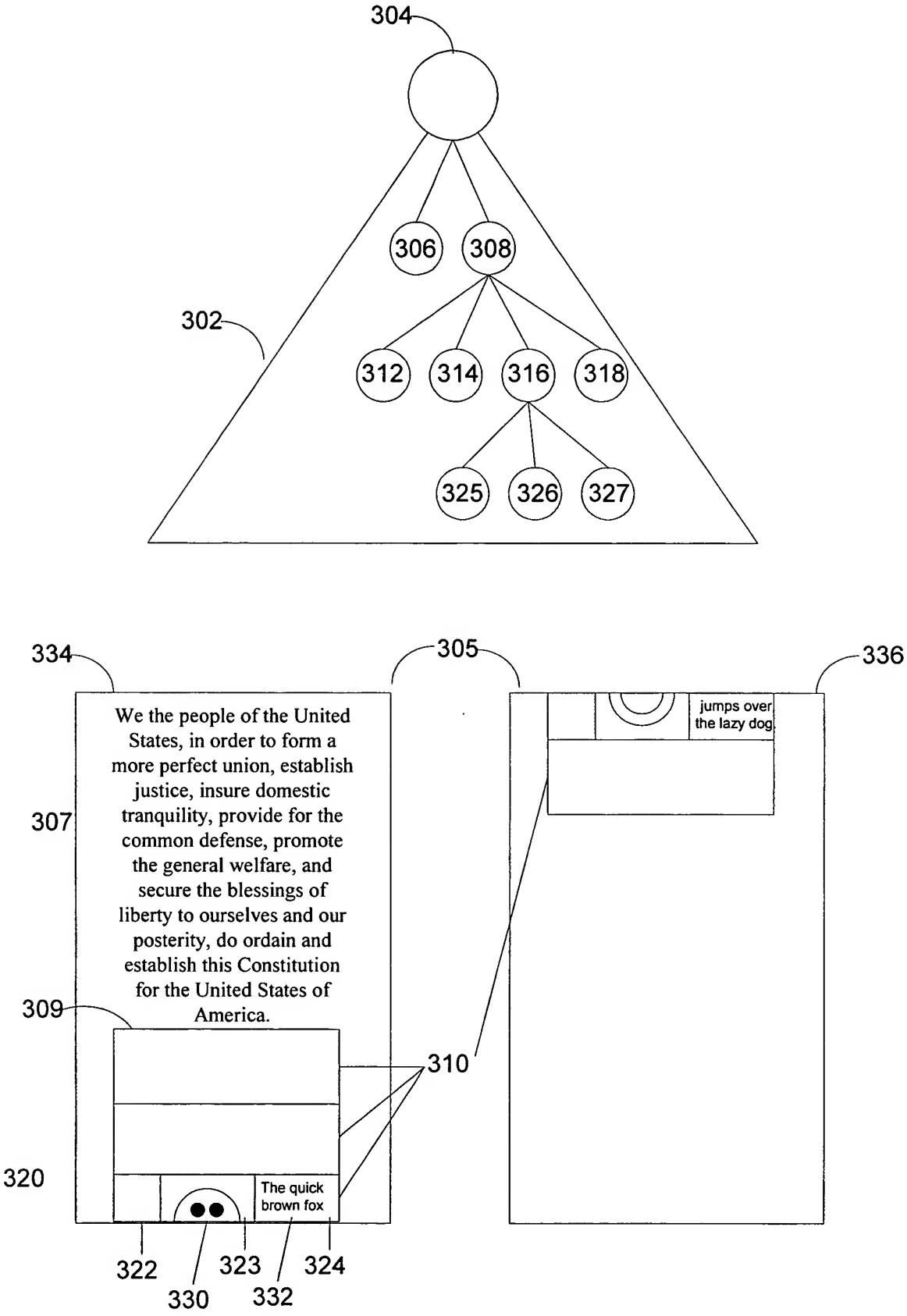

304
306
308
302
312
314
316
318
325
326
327
334
305
336
We the people of the United States, in order to form a more perfect union, establish justice, insure domestic tranquility, provide for the common defense, promote the general welfare, and secure the blessings of liberty to ourselves and our posterity, do ordain and establish this Constitution for the United States of America.
307
309
310
320
The quick brown fox
jumps over the lazy dog.
322
330
323
332
324

FIG. 3

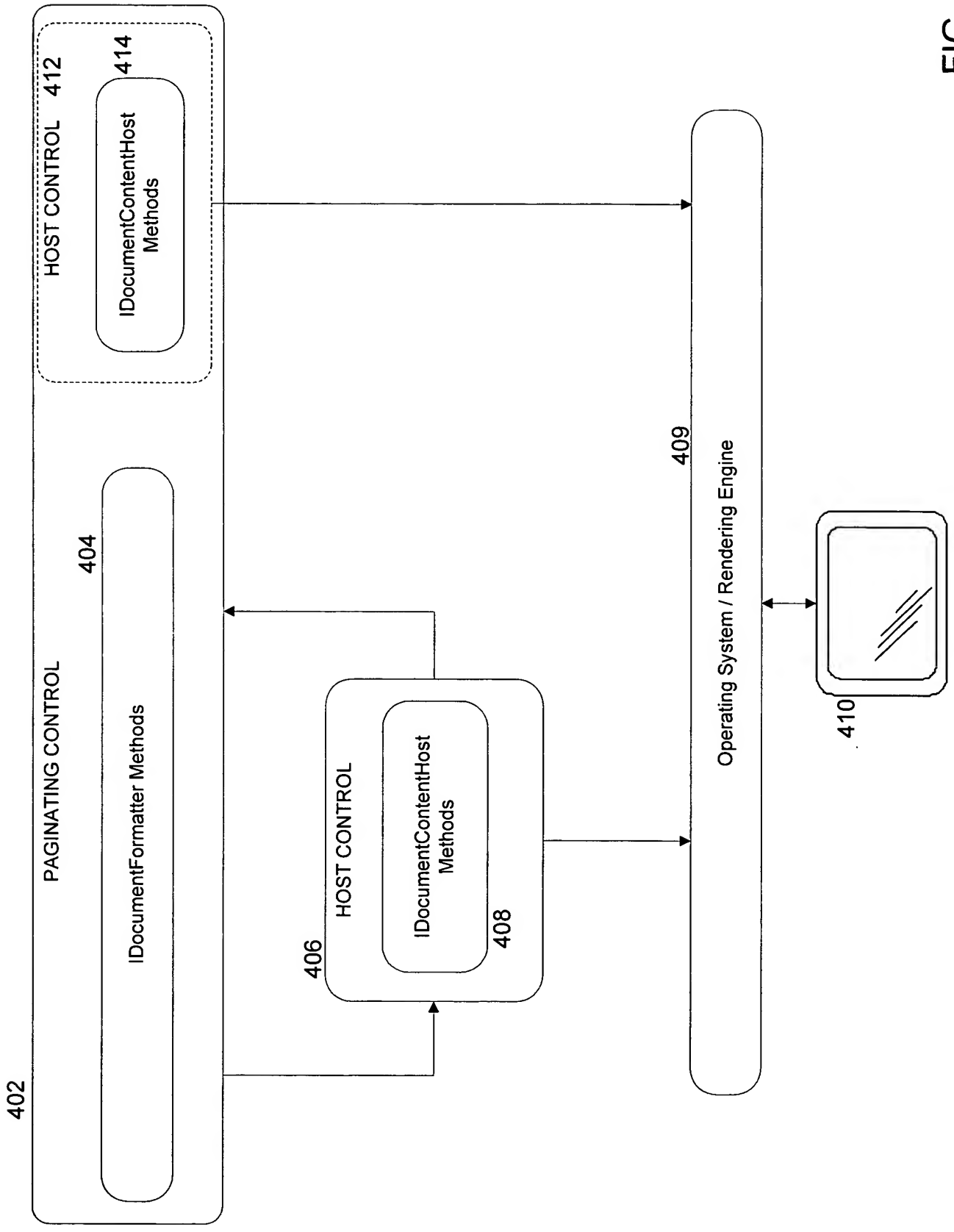

402
PAGINATING CONTROL
404
IDocumentFormatter Methods
HOST CONTROL 412
414
IDocumentContentHost Methods
406
HOST CONTROL
IDocumentContentHost Methods
408
409
Operating System / Rendering Engine
410

FIG. 4

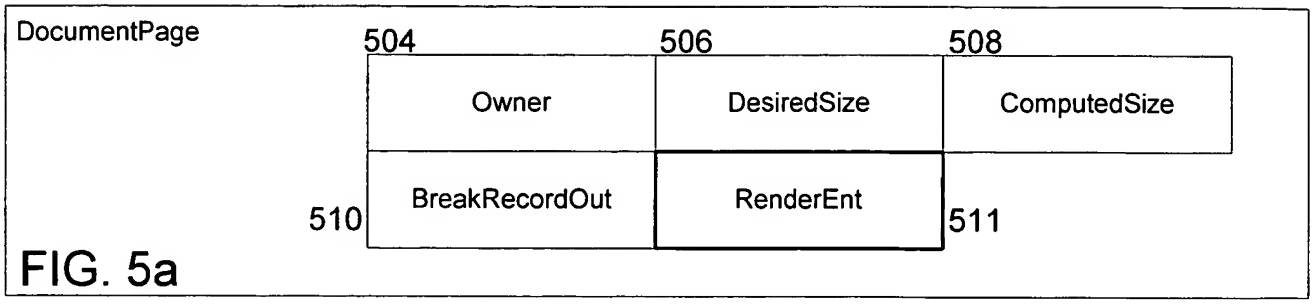


FIG. 5a

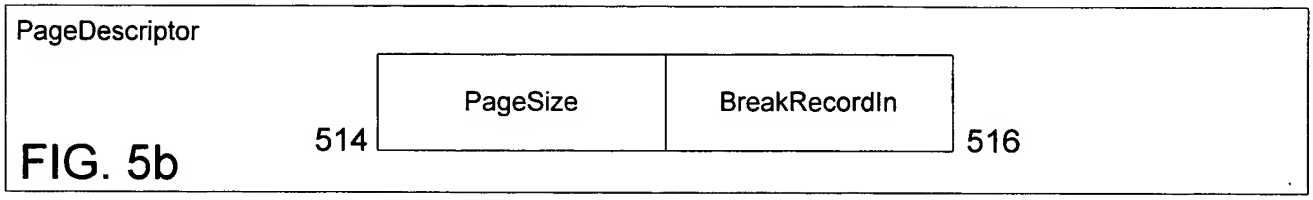


FIG. 5b

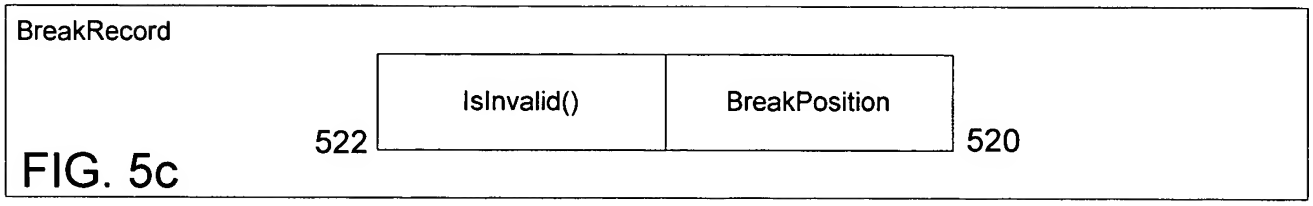


FIG. 5c

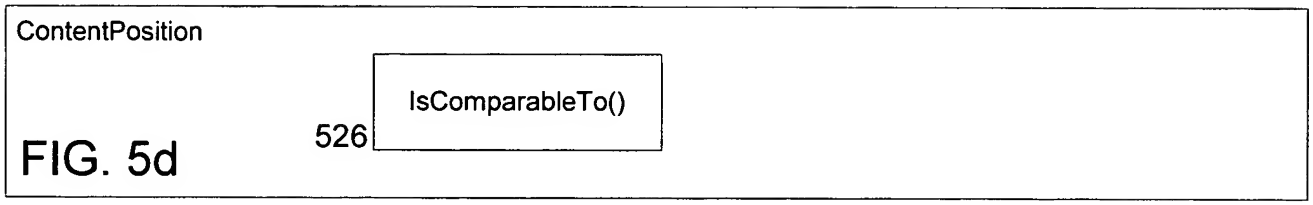


FIG. 5d

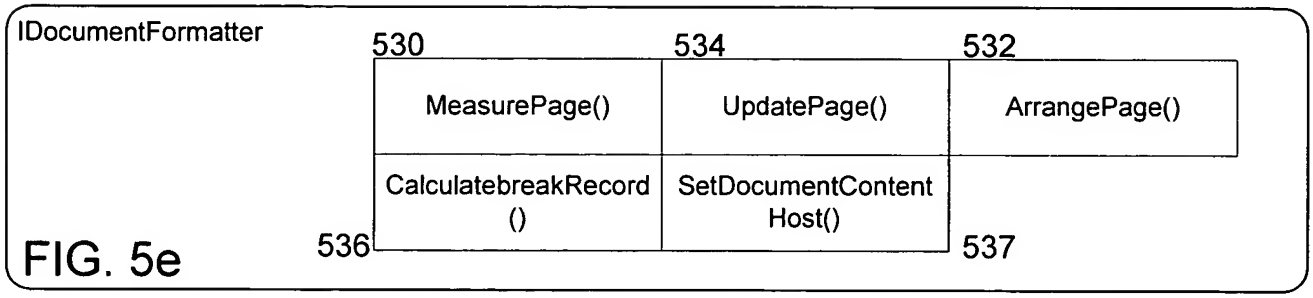


FIG. 5e

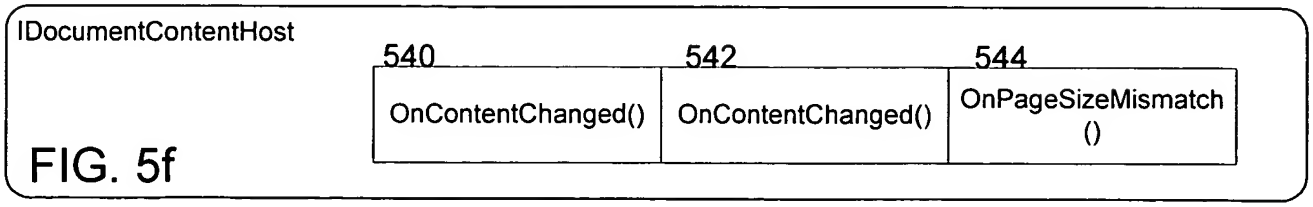


FIG. 5f

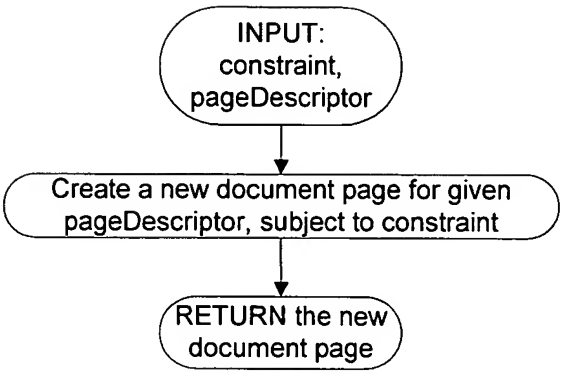


FIG. 6a

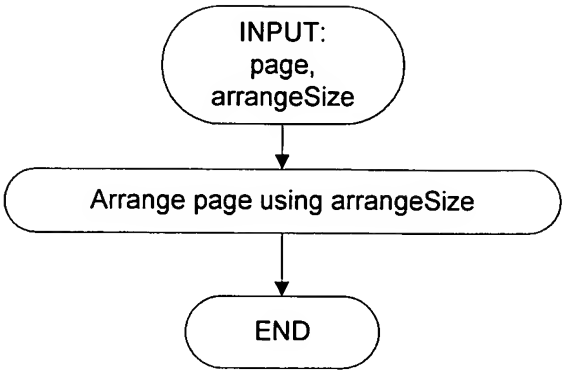


FIG. 6b

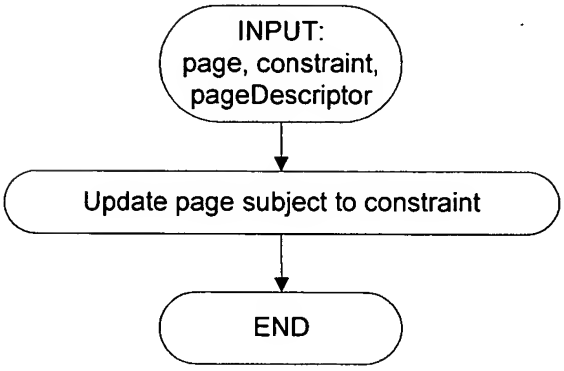


FIG. 6c

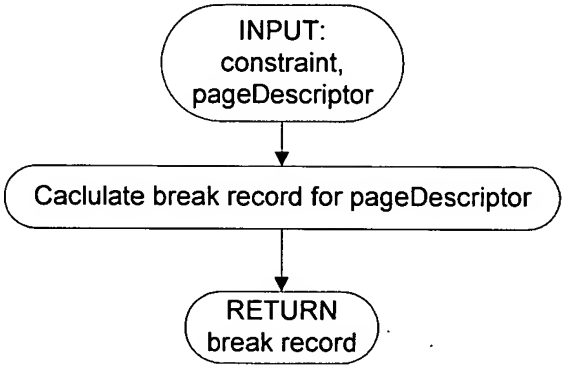


FIG. 6d

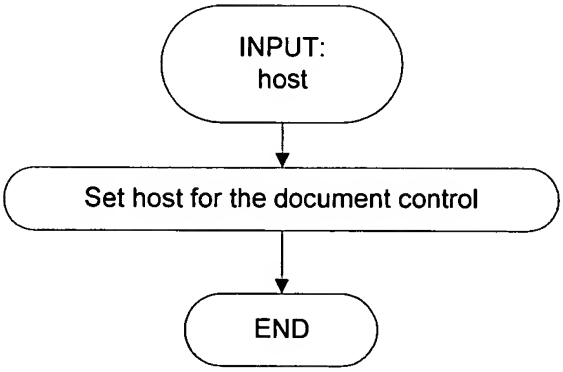


FIG. 6e

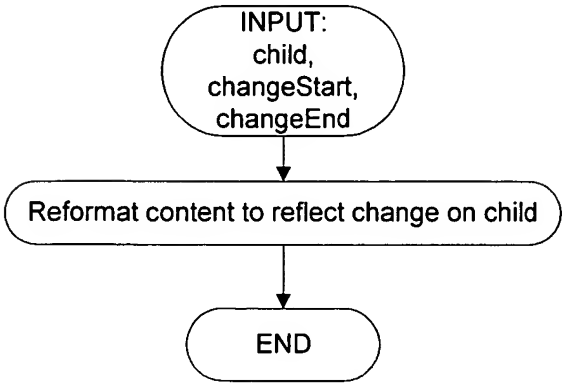


FIG. 7a

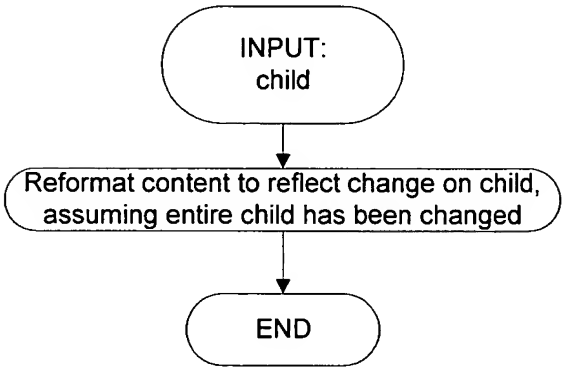


FIG. 7b

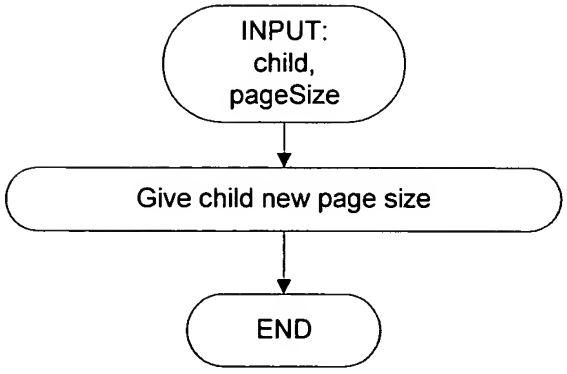


FIG. 7c

```
interface IDocumentContentHost
{
  /// <summary>
  /// Invoked on the host by one of its children (IDocumentFormatter
  /// object), when content is changing. The host needs to reformat
  /// at least part of its content to reflect changes on the child.
  /// </summary>
  /// <param name="child">Child which has been changed.</param>
  /// <param name="changeStart">Start position of a change.</param>
  /// <param name="changeEnd">End position of a change.</param>
  void OnContentChanged(
    IDocumentFormatter child,
    ContentPosition changeStart,
    ContentPosition changeEnd);

  /// <summary>
  /// Invoked on the host by one of its children (IDocumentFormatter
  /// object) when content is changing. The host needs to reformat
  /// at least part of its content to reflect changes on the child.
  /// Assume that entire child has been changed.
  /// </summary>
  /// <param name="child">Child which has been changed.</param>
  void OnContentChanged(
    IDocumentFormatter child);

  /// <summary>
  /// Invoked on the host by one of its children (IDocumentFormatter
  /// object) when content has explicit page size requirement, which
  /// is not matching the current page size (in PageDescriptor).
  /// </summary>
  /// <param name="child">Child which requires new page size.</param>
  /// <param name="pageSize">Page size.</param>
  void OnPageSizeMismatch(
    IDocumentFormatter child,
    Size pageSize);
}
```

FIG. 8

```
interface IDocumentFormatter
{
  /// <summary>
  /// Creates and formats a new document page.
  /// Throws exception if host is not set.
  /// </summary>
  /// <param name="constraint">Size constraint for the page.</param>
  /// <param name="pageDescriptor">Page descriptor; always required.
  /// For bottomless scenarios height is always = ∞.</param>
  /// <returns>An object representing formatted page.</returns>
  DocumentPage MeasurePage(
    Size constraint,
    PageDescriptor pageDescriptor);

  /// <summary>
  /// Update existing document page.
  /// Throws exception if host is not set.
  /// </summary>
  /// <param name="page">DocumentPage to update.</param>
  /// <param name="constraint">Size constraint for the page.</param>
  /// <param name="pageDescriptor">PageDescriptor; always required.
  /// For bottomless scenarios height is always = ∞.</param>
  void UpdatePage(
    DocumentPage page,
    Size constraint,
    PageDescriptor pageDescriptor);

  /// <summary>
  /// Arranges existing document page.
  /// Throws exception if host is not set.
  /// </summary>
  /// <param name="page">DocumentPage to arrange.</param>
  /// <param name="arrangeSize">Desired size of the page.</param>
  void ArrangePage(
    DocumentPage page,
    Size arrangeSize);

  /// <summary>
  /// Calculates only break record for the document page.
  /// Throws exception if host is not set.
  /// </summary>
  /// <param name="constraint">Size constraint for the page.</param>
  /// <param name="pageDescriptor">Page descriptor.</param>
  /// <returns>Page break record.</returns>
  BreakRecord CalculateBreakRecord(
    Size constraint,
    PageDescriptor pageDescriptor);

  /// <summary>
  /// Set the host of the document control.
  /// </summary>
  /// <param name="host">Host object, which receives notifications
  /// about content changes.</param>
  void SetDocumentContentHost(
    IDocumentContentHost host);
}
```

FIG. 9

```
abstract class DocumentPage
{
  /// <summary>
  /// Object which DocumentPage is representing.
  /// </summary>
  Object Owner { get; }

  /// <summary>
  /// Desired size of the page.
  /// </summary>
  Size DesiredSize { get; }

  /// <summary>
  /// Computed size of the page.
  /// </summary>
  Size ComputedSize ( get; )

  /// <summary>
  /// BreakRecord indicating break position of the page.
  /// 'null' if the page is bottomless or it is the last page.
  /// </summary>
  BreakRecord BreakRecordOut { get; }

  /// <summary>
  /// RenderableEntity node representing content of the page.
  /// </summary>
  RenderableEntity RenderEnt { get; }

  /* Detailed interface for document related features                */
  /* (footnotes, pagenotes, etc.) to be provided                     */
}
```

FIG. 10a

```
sealed class PageDescriptor
{
  PageDescriptor(
    Size pageSize,
    BreakRecord breakRecordIn);

  /// <summary>
  /// Size of the document page (top-level page).
  /// </summary>
  Size PageSize { get; }

  /// <summary>
  /// BreakRecord of the previous page. Necessary to continue
  /// page formatting.
  /// </summary>
  BreakRecord BreakRecordIn { get; }
}
```

FIG. 10b

```
abstract class BreakRecord : IComparable
{
  /// <summary>
  /// Determines if the break record is invalidated by specified
  /// dirty text range (DTR).
  /// </summary>
  /// <param name="changeStart">Start position of a change.</param>
  /// <param name="changeEnd">End position of a change.</param>
  /// <returns>'true' if is invalid, 'false' otherwise.</returns>
  bool IsInvalid(
    ContentPosition changeStart,
    ContentPosition changeEnd);

  /// <summary>
  /// Content break position. If content breaks in more than one
  /// place, this position points to further most break position.
  /// </summary>
  ContentPosition BreakPosition { get; }
}
```

FIG. 11a

```
abstract class ContentPosition : IComparable
{
  /// <summary>
  /// Return true if this instance can be compared to the passed
  /// position.
  /// </summary>
  public abstract bool IsComparableTo(
    ContentPosition position);
}
```

FIG. 11b

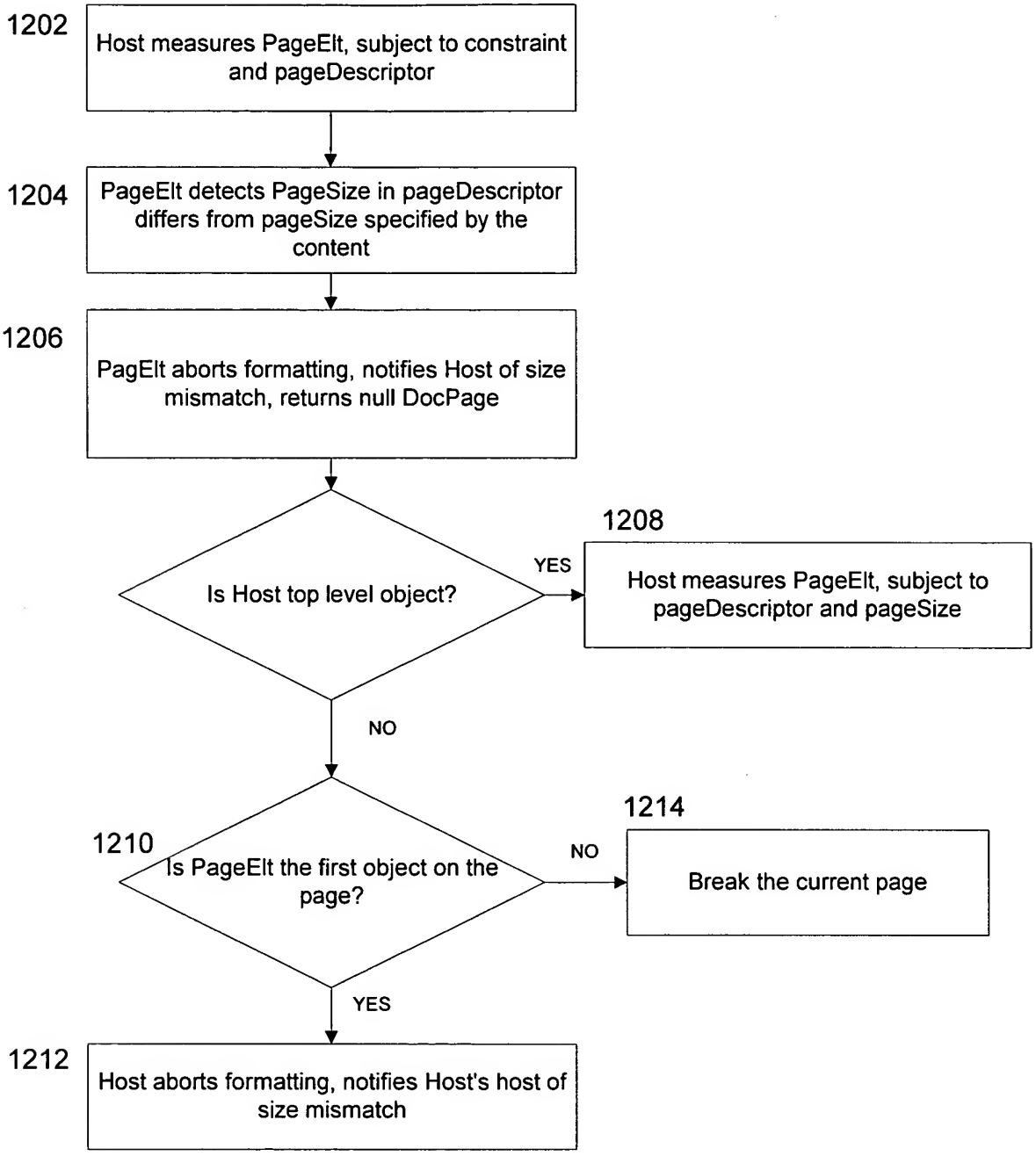

1202
Host measures PageElt, subject to constraint and pageDescriptor
1204
PageElt detects PageSize in pageDescriptor differs from pageSize specified by the content
1206
PagElt aborts formatting, notifies Host of size mismatch, returns null DocPage
Is Host top level object?
YES
1208
Host measures PageElt, subject to pageDescriptor and pageSize
NO
1210
Is PageElt the first object on the page?
NO
1214
Break the current page
YES
1212
Host aborts formatting, notifies Host's host of size mismatch

FIG. 12

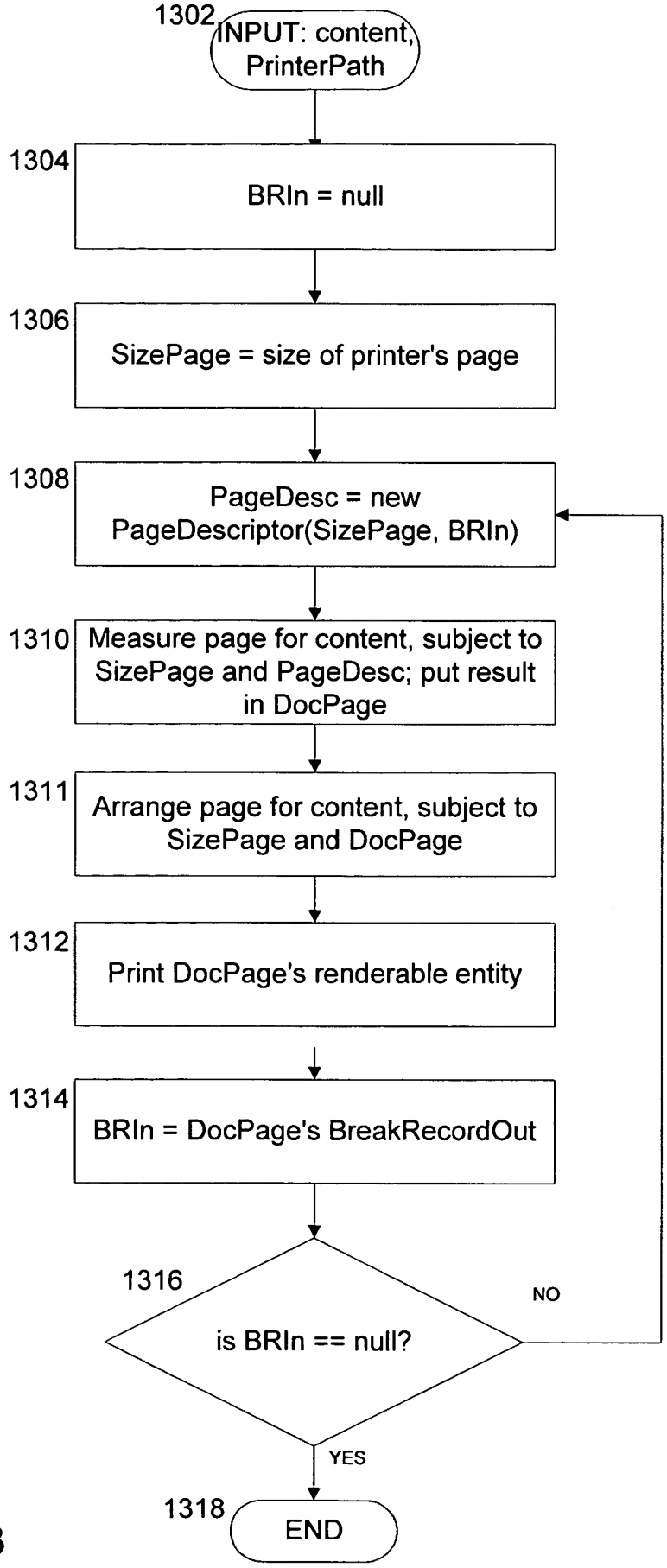

1302
INPUT: content, PrinterPath
1304
BRIn = null
1306
SizePage = size of printer's page
1308
PageDesc = new PageDescriptor(SizePage, BRIn)
1310
Measure page for content, subject to SizePage and PageDesc; put result in DocPage
1311
Arrange page for content, subject to SizePage and DocPage
1312
Print DocPage's renderable entity
1314
BRIn = DocPage's BreakRecordOut
1316
is BRIn == null?
NO
YES
1318
END

FIG. 13

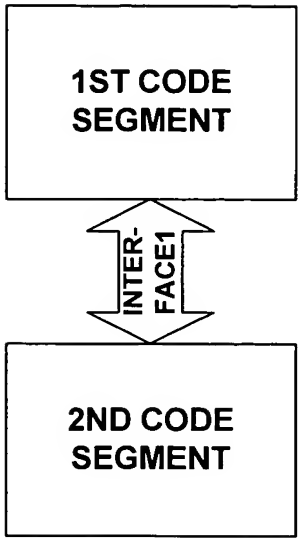


FIGURE A1

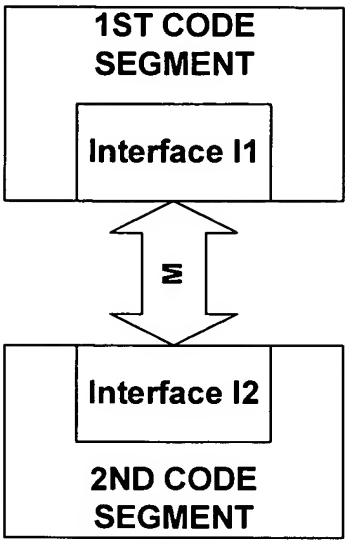


FIGURE A2

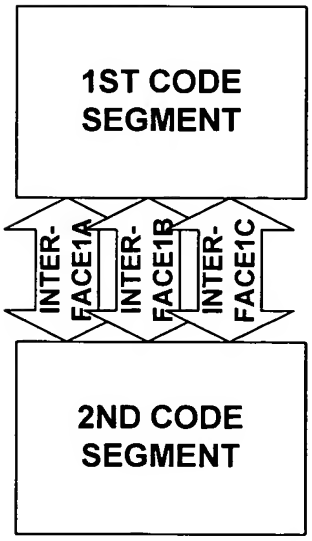


FIGURE B1

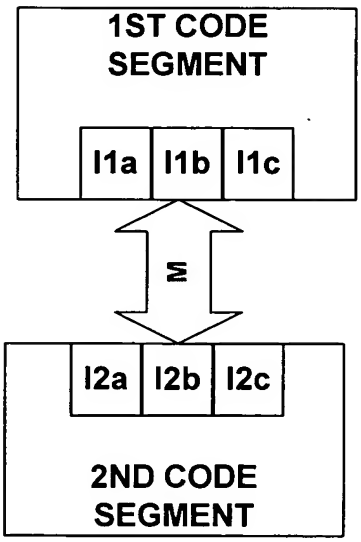


FIGURE B2

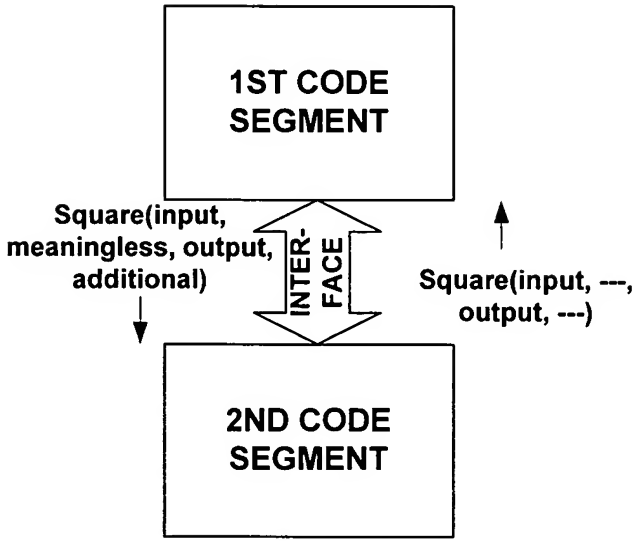


FIGURE C1

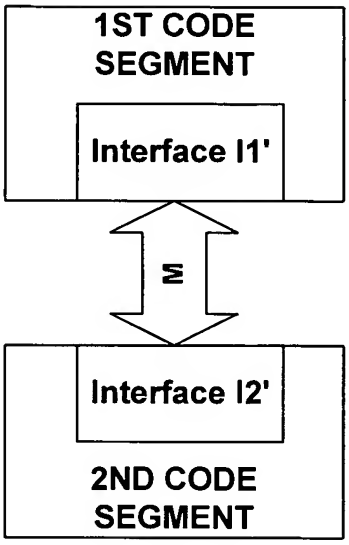


FIGURE C2

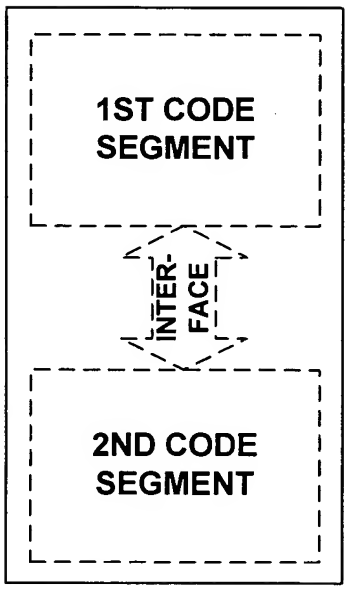


FIGURE D1

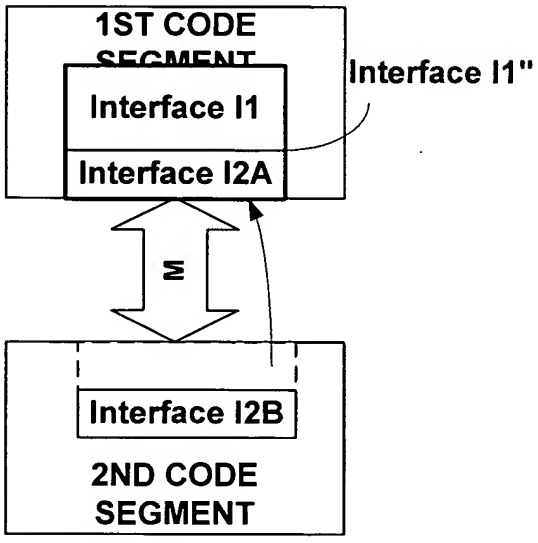


FIGURE D2

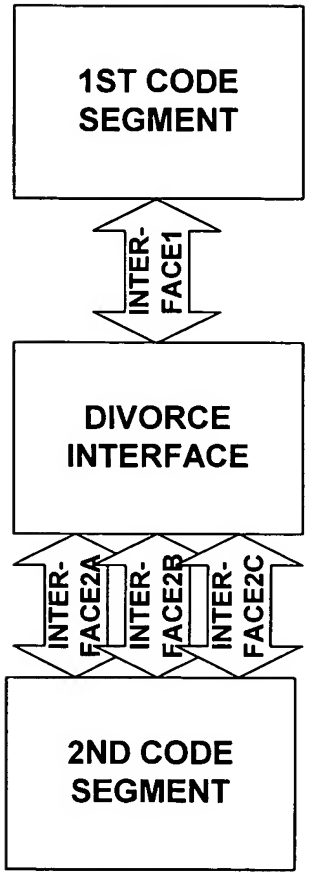


FIGURE E1

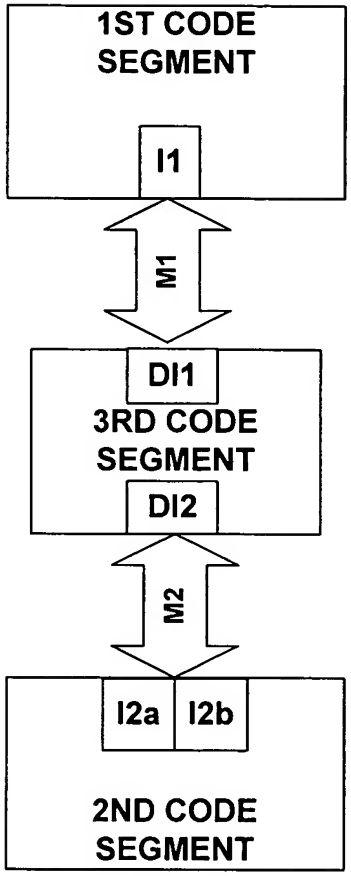


FIGURE E2

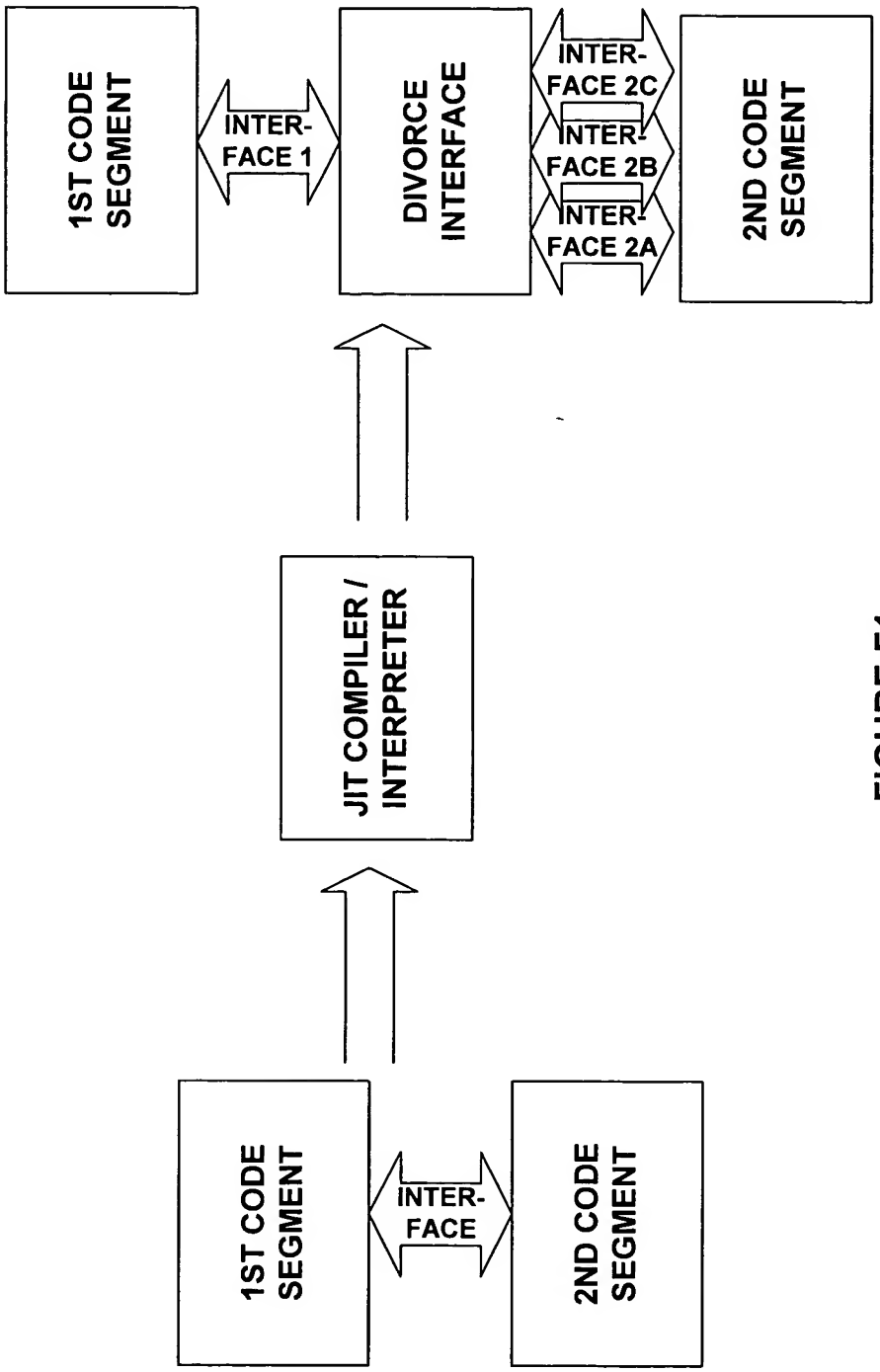

1ST CODE SEGMENT
INTER-FACE 1
DIVORCE INTERFACE
INTER-FACE 2C
INTER-FACE 2B
INTER-FACE 2A
2ND CODE SEGMENT
JIT COMPILER / INTERPRETER
1ST CODE SEGMENT
INTER-FACE
2ND CODE SEGMENT

FIGURE F1

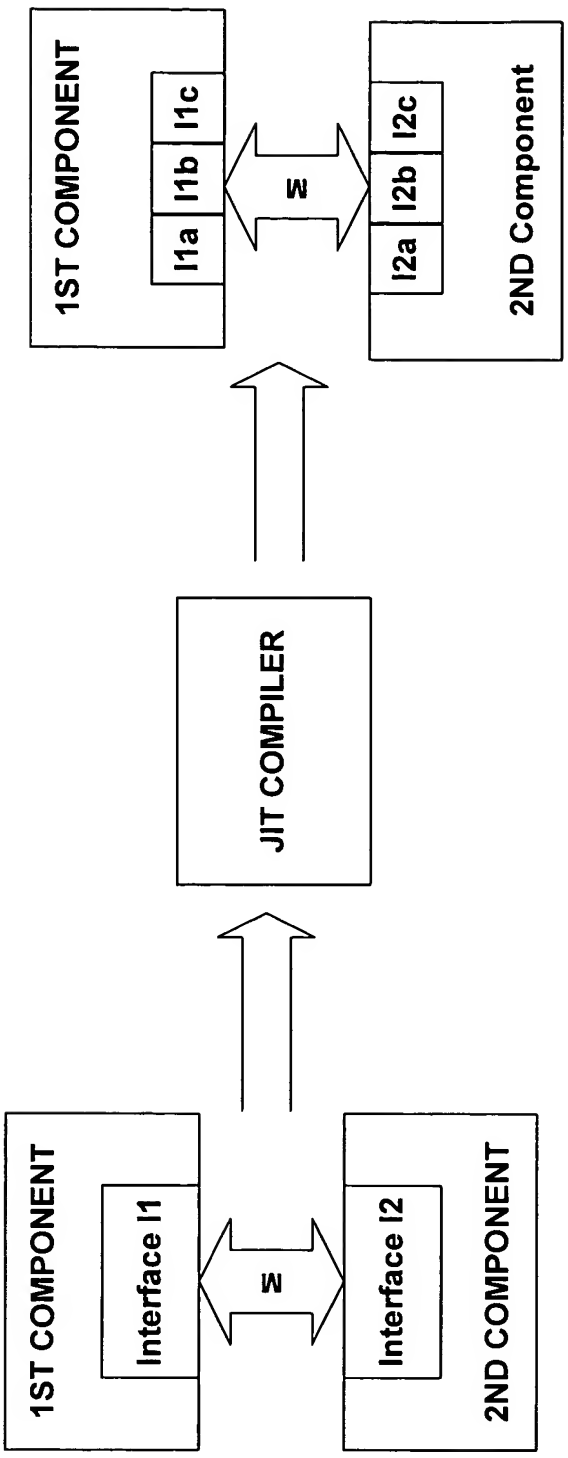

1ST COMPONENT
I1a
I1b
I1c
M
I2a
I2b
I2c
2ND Component
JIT COMPILER
1ST COMPONENT
Interface I1
M
Interface I2
2ND COMPONENT

FIGURE F2